Extractor:支持查询重构的高效数据库关键词检索系统

王新军,闫 实,彭朝晖,李庆忠
(山东大学计算机科学与技术学院,山东济南 250101)

摘 要: 数据库关键词检索由于具有简便易用的特点,成为数据处理中的一项关键技术和研究热点.目前已有的技术还存在着时间复杂度高、检索结果不够精准等问题.针对存在的问题,本文建立了基于关系数据库的关键词检索系统 Extractor.Extractor 改进了传统的数据图结构,提出了新的 top-k 结果树生成和排序机制,提出了基于词关系的查询重构方法.实验证明 Extrator 具有较好的检索效果和较高检索效率.

关键词: 关键词检索; 关系数据库; 查询重构; top-k 检索
中图分类号: TP311 **文献标识码:** A **文章编号:** 0372-2112 (2014)02-0209-08
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2014.02.001

Extractor:A Query-Reformulation Embedded Efficient Keyword Search System over Relational Databases

WANG Xin-jun, YAN Shi, PENG Zhao-hui, LI Qing-zhong
(*School of Computer Science and Technology, Shandong University. Jinan, Shandong* 250101, *China*)

Abstract: Keyword Search over Relational Databases (KSORD), a more straightforward and user-friendly way of retriving data in relational databases, has raised much attention and been widely studied in recent years. However, there are still some problems in the existing techniques. To solve these problems, Extractor, a new KSORD system based on datagraph, is proposed in this paper. The datagraph used by Extractor is derived from exiting datagraphs, which has been improved a lot and the new features are the bases of the new generating and ranking mechanism of result trees used in Extracor. The interactive query-reformulation method based on word co – occurence is also embedded in Extractor to help users find the right query. Experiments verify the effectiveness and efficiency of Extractor.

Key words: keyword search; relational database; query reformulation; top-k query

1 引言

近年来,随着大规模数据处理技术在各行各业的广泛应用,关系数据库作为目前应用最为广泛的数据库,越来越多的人成为了关系数据库的使用者.然而,传统关系数据库只允许用户使用 SQL 查询语言进行查询.而目前广泛使用的互联网中的信息检索(Information Retrieval,简称 IR),则采用了另一种完全不同风格的内容检索方式,即关键词查询,为广大用户所熟悉.随着应用需求的驱动,关系数据库关键词检索(Keyword Search Over Relational Databases,简称 KSORD)得到不断发展[1~3].KSORD 使得用户不再需要了解数据库模式和使用 SQL 语言,只需要使用关键词就可以检索关系数据库,给用户带来了便利.目前关于 KSORD 已经有许多研究工作,包括 DBXplore[4], DISCOVER[5], IR-Style[6], BANKS[7,8], SEEKER[9]等.

KSORD 的检索目标是要发现关键词之间的语义关系,主要的研究内容包括:

(1)如何组织信息碎片:用户需要的结果可能不只来自一个元组,而是以碎片的形式散落在相互关联的若干个元组当中.这时,KSORD 要做的工作不仅是在数据库中收集相关的碎片,而且要发现这些碎片之间存在的关系,并按照所有可能的情况重新拼装这些碎片作为搜索的结果.文献[10]把这种问题称为"结构化关键词查询",而把数据库自身提供的针对单文本属性的搜索,称为"全文关键词查询".

收稿日期:2012-12-14;修回日期:2013-08-30;责任编辑:蓝红杰
基金项目:国家自然科学基金(No.61170052;No.61100167);国家科技支撑计划(No.2012BAH54F04);山东省自然科学基金(No.ZR2010FM033)

(2)如何排序相关的结果:由于系统会根据数据库的结构和信息碎片的分布重组各种可能的结果,一个用户查询可能产生了大量的结果.KSORD 要将结果按照某种顺序展示出来,排序的原则就是:用户越对某一结果感兴趣,这个结果就越排得靠前.

(3)在查询关键词不能确切反映用户需求的时候,如何帮助用户重构查询:关键词查询将用户从繁复的 SQL 语句中解放出来,但在很多情况下,用户难以用寥寥数词准确表达自己的需求,这就要求系统采取某种方式实现与用户的互动,帮助用户正确表达需求.

在当前的相关研究中,许多工作使用图来表示数据库中结构化数据之间关系,从而把信息碎片的组织转变为图上的搜索问题,使用数据图的代表性研究包括 BANKS、ObjectRank[10]、DPBF[11]等,文献[12,13]研究了图的搜索技术.查询重构在 IR 领域得到较多关注[14],但对于数据库关键词检索,目前的研究成果并不多见,文献[15]提出一种基于相关反馈和伪反馈的查询重构方法,相对还比较初步.

2 系统框架和数据图

2.1 Extractor 系统框架

如图 1 所示,Extractor 系统的工作流程分为三个模块:系统启动和数据库的预处理、top-k 结果的产生以及查询的重构.

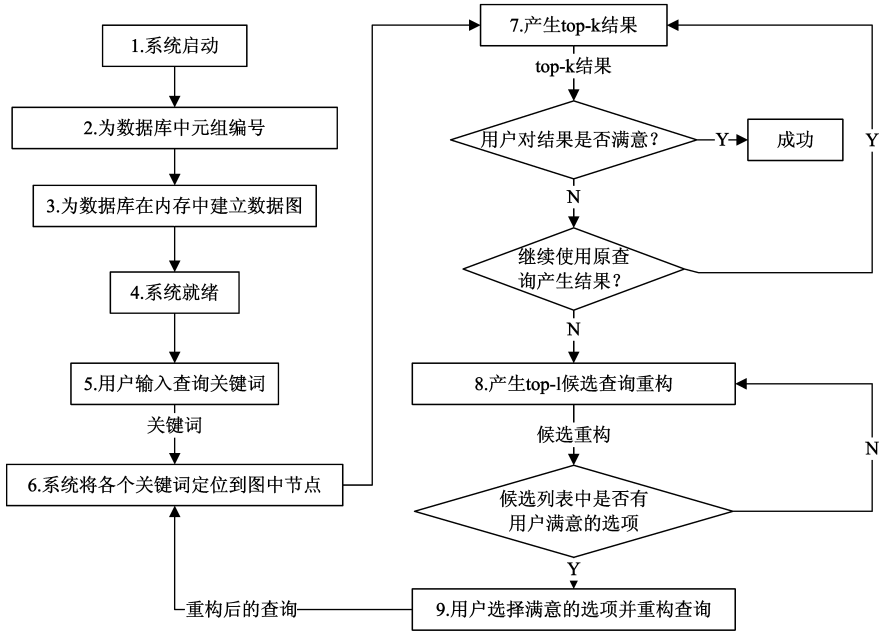


图1 Extractor系统工作流程

系统启动后,会遍历整个数据库,完成初始化工作,包括为数据库元组编号、为元组在内存中的数据图建立对应节点和相关边.

当系统收到用户提交的原始查询后,会首先根据原始查询产生 K 个用户最有可能感兴趣的结果,并展示给用户,如果用户找到了需要的信息,查询结束,否则用户选择继续产生结果或者进行查询重构.

如果用户选择继续查询,系统会在用户尚未看到的结果中产生 K 个用户最有可能感兴趣的结果.

如果用户选择查询重构,系统会根据数据库中词的关联情况产生 L 个和原始查询关联度最高的词,用户可以选择其中的一个或几个加入原始查询,构建新的查询.因此在 Extractor 中,用户输入的原始查询不必非常详细,可以是只有一两个词,然后由系统引导用户输入更精确的关键词,得到满意的结果.

2.2 无向加权数据图定义

Extractor 的 top-k 结果产生和查询重构都是基于本文定义的无向加权数据图,数据图是使用节点代表元组,使用边代表元组之间的关联,本文的数据图是在 BANKS[7,8]的数据图基础上进行了改进.

2.2.1 边的权值

在边的权值的计算上,Extractor 采用了一个新的方法计算权值,如公式(1)所示:

$$W_{e_{u,t}} = W_R(e_{u,t}) / \left(1 + \sqrt{N(v_u, e_{u,t}) \cdot N(v_t, e_{u,t})}\right) \tag{1}$$

$W_{e_{u,t}}$ 为一条节点 v_u 和节点 v_t 之间的无向边的权重.$W_R(e_{u,t})$ 为边 $e_{u,t}$ 表示的数据库中某个主外码联系的预设相关性基数($0 < W_R(e_{i,j}) < 1$,可默认为 1),$N(v_u, e_{u,t})$,$N(v_u, e_{u,t})$ 分别代表节点 v_u,v_t 上与边 $e_{u,t}$ 性质相

同的边的数量.所谓的性质相同,是指把代表同样主外码联系的边看作具有相同性质.

公式(1)中可以看出,边的权值被定义为两个节点上与(u,v)性质相同边的数量的所属平均值的倒数,相比较已有的数据图模型,Extractor 的数据图模型细化了边的种类,分别计算了每种边的出入度,避免混淆,将边的两个节点上该类型边的出入度都考虑在内,关联度的判断更加精确.

2.2.2 节点的权值

节点的权值分为节点自身权值 W_{v_i}和与关键词相关程度权值$W_{v_i,p}$.节点自身权值 W_{v_i}可以看作是这个节点的默认受关注程度:如果一个节点和其他节点联系越多,可以认为这个节点的受关注程度越高. Extractor 使用节点 v_i 自身、v_i 的相邻节点以及 v_i 相邻节点的相邻节点的出度入度之和的某种关系来表示 W_{v_i},本文称之为三阶加权出入度.计算公式如下:

$$R_{v_i}=\sum_{v_j\in V_{v_i}} W_I(e_{i,j}) \tag{2a}$$

$$R'_{v_i}=\sum_{v_j\in V_{v_i}} W_I(e_{i,j})\cdot\sqrt{R_{v_j}} \tag{2b}$$

$$W_{v_i}=R''_{v_i}=\sum_{v_j\in V_{v_i}} W_I(e_{i,j})\cdot\sqrt{R'_{v_j}} \tag{2c}$$

公式(2a),(2b),(2c)分别表示节点 v_i 的一阶、二阶、三阶加权出入度.式中 V_{v_i}表示与节点 v_i 直接相连的边的集合,$W_I(e_{i,j})$为边 $e_{u,t}$表示的数据库中某个主外键联系的预设影响度基数$(0<W_I(e_{i,j})\le 1$,可默认为 1,同一种主外键关系的 $W_I(e_{i,j})$与上文中的 $W_I(e_{i,j})$可以被设定为不同的值).根据以上三式可以看出,一阶加权出入度(2a)表示节点自身的加权出入度,二阶加权出入度(2b)表示与节点相邻的节点的加权出入度的开平方之和,三阶加权出入度(2c)表示与节点相邻的节点的二阶加权出入度的开平方之和.

2.2.3 与关键词相关程度权值

节点 v_i 的权值 $W_{v_i,0}$为节点和整个查询语句的相关程度. $W_{v_i,1}$到 $W_{v_i,n}$(其中 n 为查询关键词的数量)分别代表 v_i 和一个关键词的相关程度.例如,查询 Q 包含 n 个关键词 k_1 到 k_n,k_p 为其中一个关键词,对于节点 v_i,$W_{v_i,p}$就表示其和关键词 k_p 的关联程度.

在查询执行前,长度为 n 的查询 Q 会被拆分成相互独立的关键词 k_1 到 k_n,然后对这些关键词执行查询.这些关键词可能是简单关键词,如"kurt","DBMS"等,也可能是复合关键词,如"Author: bonny | bonnie | bonie",但这不影响关键词与节点的匹配.在 Extractor 中,对于关键词 k_p,无论 k_p 是简单关键词还是复合关键词,如果节点 v_m 与之匹配,则令 $W_{v_m,p}$为 1.而对于不和 k_p 直接匹配的节点v_i,$W_{v_m,p}$需要根据其相邻节点的$W_{v_j,p}$计算得出.$W_{v_m,p}$的计算公式如下:

$$W_{v_j,p}=1,\quad v_i\in V_p \tag{3a}$$

$$W_{v_j,p}=\max(W_{v_j,p}\cdot W_{e_j,j}),v_i\notin V_p,v_j\in V_{v_i} \tag{3b}$$

在式(3a)和(3b)中,V_p 表示关键词 k_p 直接匹配的节点的集合,V_{v_i}表示所有与 v_i 邻接的节点的集合,$W_{e_i,j}$表示连接v_i 和v_j 的边 $e_{i,j}$的权值.在这里,可以把 $W_{e_i,j}$看作是节点 v_j 与节点 v_i 的关联度,而 $W_{v_j,p}\cdot W_{e_{i,j}}$就是节点 v_j 能给予节点 v_i 关于 k_p 的关联度,那么,对于不和 k_p 直接匹配的节点 v_i,$W_{v_j,p}$的值就是其从相邻节点那里获得的与k_p 的关联度的最大者.例如图 2 中,假设节点 v_{15} 与关键词 k_p 不直接匹配,那么其与 k_p 的相关度 $W_{v_{15},p}$ 就等于 $\max(0.09,0.12,0.1,0.16)$,最终的结果就是 0.16.

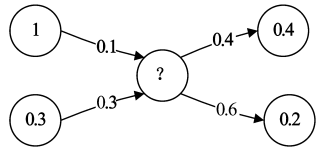


图2　与关键词相关程度权值的计算

当所有节点与所有关键词的相关度计算完毕后,Extractor为每个数据图节点计算其与整个查询语句的相关程度 $W_{v_i,0}$,$W_{v_i,0}$为节点和所有关键词相关程度$W_{v_i,i}$的乘积,即

$$W_{v_i,0}=\prod_{p}^{n} W_{v_i,p} \tag{4}$$

其中 p 表示查询包含的关键词的数量,可以看出,如果任何节点和任何一个关键词无关联,那么认为它和整个查询无关联.

通过上面介绍的两个权值计算节点的总权值 W'_{v_t},公式如下:

$$W'_{v_t}=W_{v_t,0}\cdot W_{v_t}^{\lambda} \tag{5}$$

其中 λ 是根据数据库的结构设定的正值,表示对节点本身的受关注程度和节点与整个查询语句的相关程度在最后的权值中占的比重.

3 Top-k 结果产生和查询重构

在 Extractor 中,采用数据库系统提供的全文索引,从数据库(数据图)中定位与关键词匹配的节点.在此基础上,计算产生 top-k 结果和进行查询重构.

3.1 Top-k 结果的产生

3.1.1 结果树的定义

在 Extractor 中,每一个结果是一个由节点组成的树,称之为结果树.结果树由核心节点、信息点、路径节点组成.

结果树:结果树是从数据图中摘取的包含所有关

键词的匹配节点的树形连接子图,包含一个核心节点,**h** 个信息节点(h 为不大于关键词的数量的自然数),以及若干个路径节点(也可以没有路径节点).图 3 就是在 DBLP[16]数据图中查询“Hristidis Databases”的结果树示例.

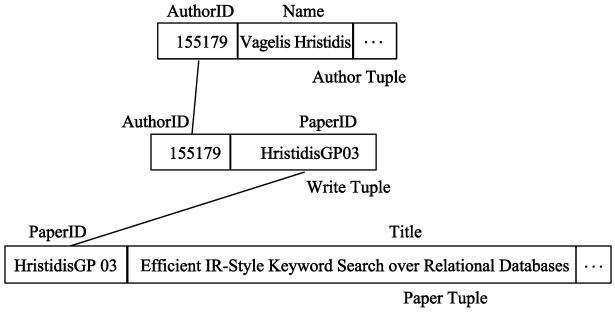


图3 结果树示例

核心节点:结果树中 W'_{v_i} 最大的节点,同时也是树中的根节点,决定了该结果树的兴趣度和信息节点,但不一定直接与关键词匹配.在图 3 中,三个节点都可以是核心节点.为了说明下面两种节点,假设 Author Tuple 为核心节点.

信息节点:与关键词匹配的节点,也就是包含信息碎片的节点.结果树中的叶节点都是这类节点,核心节点也可以同时是信息节点.在图 3 中,节点 Author Tuple 和 Paper Tuple 因为分别包含“Hristidis”和“Databases”,所以为信息节点.

路径节点:核心节点到信息节点路径上的节点.如图 3 中的 Write Tuple,虽然不包含信息,但却是结果树不可或缺的组成部分.

3.1.2 Top-k 结果的产生

Top-k 即是要产生 k 个用户最可能感兴趣的结果.首先要对所有 W'_{v_i} 不为零的节点建立集合 V_Q,并在 V_Q 中找到 W'_{v_i} 最大的节点 v_{Q1},以 v_{Q1} 为核心节点创建结果树 T_1.创建完毕后,重新在 V_Q 中还未进入结果树的节点中寻找 W'_{v_i} 最大的节点 v_{Q2},并以之为核心创建结果树 T_2.循环往复,直到产生 k 个结果树.

如果 v_{Qt} 的关键词影响源与已创建的任何结果树不相同.则可以以 v_{Qt} 为核心节点创建结果树,以 $W'_{v_{Qt}}$ 为结果树的兴趣度,以 v_{Qt} 的影响源为结果树的信息节点,v_{Qt} 到其影响源的途经节点为结果树的路径节点;这些节点一起就构成了一棵结果树.因为 v_{Qt} 到 v_{Qk} 是按照 W'_{v_t} 从大到小的顺序排列的,先生成的结果树的兴趣度也就大于后生成的.因此,已生成的结果树的兴趣度必然大于未生成的,结果树生成的顺序也就是 top-k 的排序.产生 top-k 结果树的算法如下:

算法 1 产生 top-k 结果树的算法

输入:数据图 dGraph,k 的值

输出:ResultTree[k]数组

```
Begin
01  创建 ResultTree[k]数组,int i = 0 表示第 i 棵树
02  遍历 dGraph 中的所有节点,找到 tWeight 不为 0 的节点组成集合 V_r
03  for( i = 0; i < k; i + + )
04      在 V_r中找到 tWeight 最大的节点 node[r]
05      ResultTree[i].seed = node[r];
06      for 查询 Q 中每个关键词
07          将 node[r]对应的祖节点和祖节点对应的路径写入 ResultTree[i]
08      return ResultTree[k]
End
```

3.2 查询的重构

3.2.1 查询扩展

用户可以选择启动查询扩展功能,Extractor 会提供给用户 L 个查询扩展词,用户再从其中选择一个或几个加入原始查询,使查询更加详细.例如,用户在 DBLP 中检索数据库关键词检索方面的论文,但输入的查询只是“keyword search”,启动查询扩展后,Extractor 会返回一组关键词帮助用户扩展查询以缩小结果范围,这时用户可以选择“relational”“database”加入查询,构成新的查询“keyword search relational database”,执行此查询得到用户满意的结果.

候选查询结果的生成过程中,本文吸取了 MDKQE[14]中 Web 检索扩展中的共现(co-occurrence)思想.MDKQE 使用 Surprise(S)表示关键词集 S 的耦合度,即 S 内各个关键词关系 $w_1,\cdots,w_r$ 的密切程度,Surprise(S)公式如下:

$$\text{Surprise}(S)=\frac{c(S)/c_t}{\prod_{i=1}^{r}c(w_i)/c_t}=\frac{c(S)}{\prod_{i=1}^{r}c(w_i)}Z\cdot c_t^{r-1} \tag{6}$$

其中,$c(w_i)$ 表示含有关键词 w_i 的文章篇数,$c(S)$ 表示同时含有 S 中所有关键词的文章篇数,c_t 表示文章的总篇数.根据公式(6),在各个关键词独立出现次数不变的情况下,同时出现的次数越多,Surprise(S)就越高,表明各个关键词之间的关系越密切.

在关系数据库中,关键词之间的关系无法用简单的同时出现次数来衡量,但可以用节点之间的关系来计算关键词之间的关系.对公式(6)作出修改,得到公式(7)

$$\text{Surprise}(S)=\frac{\sum_{v_i\in V}W_{v_i,S}/c_t}{\prod_{i=1}^{r}\left(\sum_{v_i\in V}W_{v_i,w_i}/c_t\right)}=\frac{\sum_{v_i\in V}W_{v_i,S}}{\prod_{i=1}^{r}\left(\sum_{v_i\in V}W_{v_i,w_i}\right)}\cdot c_t^{r-1} \tag{7}$$

在公式(7)中,V 表示所有节点的集合,W_{v_i,w_i} 表示节点 v_i 和关键词 w_i 的相关度,$W_{v_i,S}$ 表示节点 v_i 和关键词集 S 的相关度.可以看出,式子用节点关键词集相关的程度代替了同时出现的次数,这样可以在关系数据库中计算关键词之间的关系.

在计算 Extractor 的候选查询扩展时,S 只包含 2 个元素,一个是原始查询 Q,一个是备选扩展 e_i,公式(7)就变为:

$$\text{Surprise}(S)=\frac{\sum_{v_i\in V}W_{v_i,S}}{\left(\sum_{v_i\in V}W_{v_i,Q}\right)\cdot\left(\sum_{v_i\in V}W_{v_i,e_i}\right)}\cdot c_t \tag{8}$$

Surprise(S)就是对备选扩展 e_i 的评分,用 $S(Q,e_i)$ 表示.

按照公式(8)计算,每次计算需遍历一次数据图,时间耗费大,而且实际的精确度要求相对较低,因此只需计算 $S(Q,e_i)$ 的近似值即可.对公式(9)做出改进:(1)假设所有关键词节点的影响能力相同,均为 I,即:

$$\sum_{v_i\in V}W_{v_i,w}=|V_w|\cdot I \tag{9a}$$

$$W_{v_i,S}=W_{v_i,Q},v_i\in V_{e_i},W_{v_i,S}=0,v_i\notin V_e \tag{9b}$$

其中 V_w 为与关键词 w 直接匹配的节点集合;(2)因为对于任意备选关键词 w,c_t 以及 $\sum_{v_i\in V}W_{v_i,Q}$ 两个值均相同,所以可以把公式(8)中的 c_t 和 $\sum_{v_i\in V}W_{v_i,Q}$ 去掉;最终得到公式(10):

$$S(Q,e)=\frac{\sum_{v_i\in V_{e_i}}W_{v_i,S}}{|V_{e_i}|} \tag{10}$$

系统将会对数据库中所有出现的词计算 $S(Q,e_i)$,并找到 $S(Q,e_i)$ 最高的 L 个词作为查询扩展候选词提供给用户(用户还可以选择筛选掉 $|V_{e_i}|$ 为 1 的词).

相比较传统的方法,这种查询扩展方法消除了常用词的干扰.传统扩展方法常常借助常用词表消除常用词影响,但是对于某些专业领域的常用词,如 DBLP 中的 computer, database,则不容易区分.该算法中由于 $S(Q,e_i)$ 的大小与词的出现次数有关系,所以总出现次数多的词较难成为候选词.

3.2.2 查询纠错

查询纠错旨在找出查询中表达语义不准确的关键词,并给用户提示,如果用户也认定此关键词有语义偏差,可以直接输入替换词或启动查询扩展.

查询纠错也采用了上文提到的共同出现思想.Extractor 会找出查询中与其他关键词关系最小的关键词.实际应用中,沿用公式(10),由于对于查询中的每个关键词,与其余部分共同组成了原始查询,所以 $\sum_{v_i\in V}W_{v_i,S}$ 为常数,因此得到公式(11):

$$\text{Surprise}(S)=\frac{1}{\left(\sum_{v_i\in V}W_{v_i,Q}\right)\cdot\left(\sum_{v_i\in V}W_{v_i,e_i}\right)} \tag{11}$$

其中目标关键词为 e_i,以 e_i 外的其他关键词为 Q,计算查询中每个关键词的 $S(Q,e_i)$,最终找到 $S(Q,e_i)$ 最小的关键词.

4 实验评估及分析

4.1 实验设置

Extractor 是在 MS-SQL Server 上开发的,采用 C # 语言编写,和数据库的接口是 ODBC.Extractor 的实验评测是在 DBLP[16] 数据集上进行,本文将 DBLP 提供的 XML 文件按照图 4 所示的模式分解成 4 个关系:AUTHOR 中有 294 062 个元组;PAPER 中有 446 409 个元组;WRITE 中有 1 000 099 个元组;CITE 中有 111 357 个元组.AUTHOR 和 PAPER 上建立了全文索引,实验在一台 2G 内存的 Core2 2.93 GHz 计算机上进行,操作系统是 Windows7.

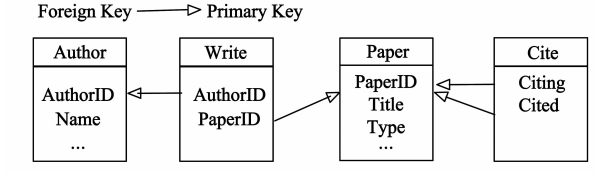


图4 DBLP数据库模式图

本文采用查准率(precision)和查全率(recall)来衡量 Extractor 的查询效果,通过计算产生结果树的数量(TreeNum)和查询消耗时间(Time)来衡量 Extractor 的查询执行效率.查准率和查全率的计算公式如下:

$$\text{Precision}=\frac{|R_a|}{|A|} \tag{12a}$$

$$\text{Recall}=\frac{|R_a|}{|R|} \tag{12b}$$

其中,$|R_a|$ 是返回的查询结果集中与查询相关的连接元组树的数量;$|A|$ 是查询结果集中的连接元组树的数量.$|R|$ 是数据库中与查询相关的连接元组树的数量.

4.2 实验结果及讨论

本实验从 BANKS[8] 和 Seeker[9] 实验所用的例子

中随机各抽取了 5 个查询作为测试用例,并转化为 Extractor 系统支持的格式,由于 Extractor 系统包含了“或”语义,所以对最后两个查询略作修改,增加了实验结果的覆盖面,如表 1 所示.为了更好的表现结果,将上述十个查询按照平均每词匹配节点 Match Points(MP)个数分为两组,MP > 2000 的为高频关键词组,MP < 2000 的为低频关键词组.

表 1 相关反馈测试用例(括号内为平均值)

查询	匹配节点(MP)
Juris Hartmanis complexity	16,4888(2452)
Bernstein concurrency	60,1185(623)
Jennifer Widom database|dbms	1,8143(4092)
Dewitt database	8,7764(3886)
Ullman data	11,17621(8816)
Krishnamurthy parametric query optimization	53,598,3133,3981(1939)
Naughton Dewitt&query processing	6,8,3133,4940(1772)
Divesh Jignesh Jagadish Timber Querying XML	1,4,4,8,610,1356(361)
Jiawei Han Data Mining	1,1301(652)
john Reid|Reed	3315,169(1742)

4.2.1 Top-k 结果的查询效果和效率分析

实验 1 关键词在数据库中的出现频率对查准率的影响

图 5 中比较的是高频关键词和低频关键词的查准率,传统的关系数据库关键词查询系统关键词出现频率对查准率有很大的影响,即在关键词出现频率较高的情况下,特别是 k 取一个较小的值时,查准率没有提升,反而下降,这影响了查询的效果,而通过对 Extractor 的测试可以看出,高频关键词组的平均查准率较低频关键词组有显著的提升,这说明在处理高频关键词上,Extractor 取得了良好的效果.

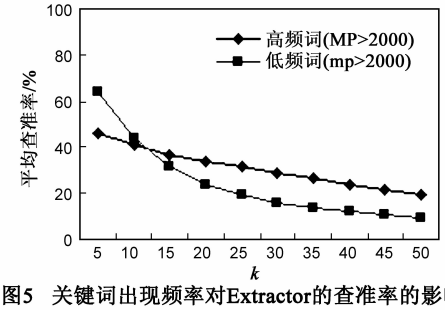


图5 关键词出现频率对Extractor的查准率的影响

实验 2 k 值对查询结果产生速度的影响

图 6 给出了对于不同 k 值,系统的时间耗费.从图中可以看出,随着 k 值的增加,对高频关键词系统执行时间也随之增加,这是产生结果树的时间耗费造成的;而对于低频词汇并不明显,是因为低频词汇的结果树总数有限,当 k 到达一定数值时会超过结果总数,系统不再产生更多的结果.但总体看来,k 的增加对总时间的影响不是很大,这给了用户对 k 值选择更大的余地.

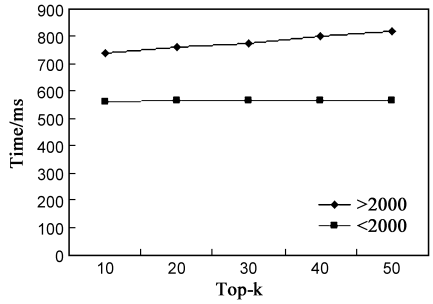


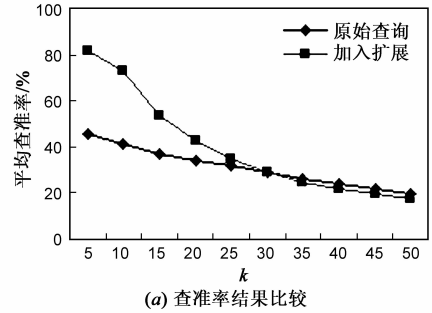
图6 Extractor的查询执行时间和k的关系

4.2.2 查询重构对查询效果的提升

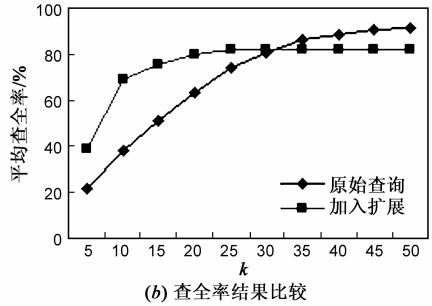
根据查询扩展和查询纠错针对的不同问题,在高频组上进行查询扩展实验,在低频组上进行查询纠错实验.

实验 3 查询扩展的应用对查询效果的提升

查询扩展前后的查准率和查全率如图 7 所示,从图中可以看出,Extractor 的查询扩展功能在 k 较小时显著的提高了查准率和查全率,但是随着 k 增加,效果越来越不明显,这是由于缩小结果范围的过程中错误的排除了一些正确结果造成的.通常来讲,这种错误排除的结果都是与查询关系较远的结果,但这在某种程度上讲也给用户造成了一定麻烦,如何降低这种错误,也是以后工作的一个重要方面.然而,即使存在这样的一些问题,对于追求较高的查询效率的用户来讲,查询扩展仍然是一个好方法.



(*a*) 查准率结果比较

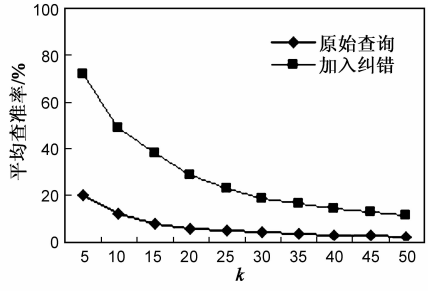


(*b*) 查全率结果比较

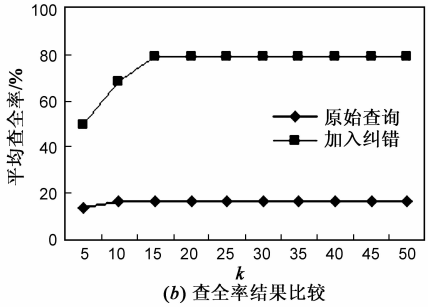
图7 查询扩展前后查准率和查全率的变化

实验 4　查询纠错的实验效果

查询纠错前后的查准率和查全率如图 8 所示,从图中可以发现,通过查询纠错,结果的查全率和查准率都有明显提高.



(*a*) 查准率结果比较



(*b*) 查全率结果比较

图8　查询纠错前后查准率和查全率的变化

4.2.3　关键词的个数对查询效率的影响

实验 5　关键词复杂程度对查询时间的影响

图 9 给出了对于不同的关键词数量,在数据图上加入关键词相关权值消耗的时间和产生结果消耗的时间.从图中可以看出,Extractor 执行查询消耗的时间是随查询的长度增长的,但是这种增长是由于查询关联的节点的增多引起的数据图构建时间增加造成的,而生成结果树的时间并未随之明显增长.

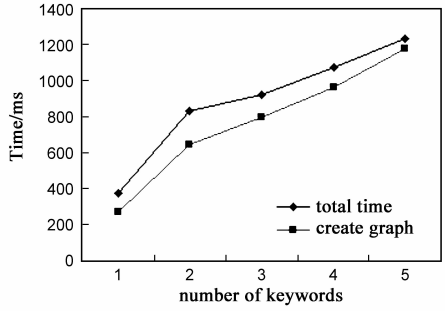


图9　关键词的数量对查询效率的影响

5　结论

本文介绍了 Extractor 数据库关键词查询系统的基本思想和实现算法.系统采用数据图为基础,研究设计新的计算方法,为数据库关键词查询中如何组织信息碎片,如何展现查询结果,以及如何交互的帮助用户重构查询提出了解决方案,并进行了实验验证.未来工作中,将会对查询语言进一步多样化,以提高查询的灵活性,另外要对查询重构的执行效率进行研究和评价.

参考文献

[1] Hulgeri A, Bhalotia G, Nakhe C, Chakrabarti S, Sudarshan S. Keyword search in databases[J]. IEEE Data Engineering Bulletin, 2001, 24(3): 22 – 31.

[2] Wang S, Zhang KL. Searchingdatabases with keywords[J]. Journal of Computer Science and Technology, 2005, 20(1): 55 – 62.

[3] 林子雨,杨冬青,王腾蛟,张东站.基于关系数据库的关键词查询[J].软件学报,2010,21(10):2454 – 2476.
Lin ZY, Yang DQ, Wang TJ, Zhang DZ. Keyword search over relational databases[J]. Journal of Software, 2010, 21(10): 2454 – 2476. (in Chinese)

[4] Agrawal S, Chaudhuri S, Das G. DBXplorer: A system for keyword-based search over relational databases[A]. Proc of the 18th Int'l Conf. on Data Engineering[C]. San Jose: IEEE Press, 2002. 5 – 16.

[5] Hristidis V, Papakonstantinou Y. DISCOVER: Keyword search in relational databases[A]. Proc of the 28th Int'l Conf on Very Large Data Bases[C]. Hong Kong: Morgan Kaufmann Publishers, 2002. 670 – 681.

[6] Hristidis V, Gravano L, Papakonstantinou Y. Efficient IR – style keyword search over relational databases[A]. Proc of the 29th Int'l Conf on Very Large Data Bases[C]. Berlin: Morgan Kaufmann Publishers, 2003. 850 – 861.

[7] Bhalotia G, Hulgeri A, Nakhe C, Chakrabarti S, Sudarshan S. Keyword searching and browsing in databases using BANKS [A]. Proc of the 18th Int'l Conf on Data Engineering[C]. San Jose: IEEE Press, 2002. 431 – 440.

[8] Kacholia V, Pandit S, Chakrabarti S, Sudarshan S, Desai R, Karambelkar H. Bidirectional expansion for keyword search on graph databases[A]. Proc of the 31st Int'l Conf on Very Large Data Bases[C]. New York: ACM 2005. 505 – 516.

[9] 文继军,王珊.SEEKER:基于关键词的关系数据库信息检索[J].软件学报,2005,16(7):1270 – 1281.
Wen JJ, Wang S. SEEKER: Keyword-Based information retrieval over relational databases[J]. Journal of Software, 2005, 16(7): 1270 – 1281. (in Chinese)

[10] Balmin A, Hristidis V, Papakonstantinou Y. ObjectRank: Authority-Based keyword search in databases[A]. Proc of the 30th Int'l Conf on Very Large Data Bases[C]. San Fransisco: Morgan Kaufmann Publishers, 2004. 564 – 575.

[11] Ding BL, Yu JX, Wang S, Qin L, Zhang X, Lin XM. Finding top-*k* min-cost connected trees in databases[A]. Proc of the 23rd Int'l Conf on Data Engineering[C]. Dallas: IEEE Com-

puter Society, 2007. 836 – 845.
[12] 高琳,覃桂敏,周晓峰.图数据中频繁模式挖掘算法研究综述[J].电子学报,2008,36(8),pp.1603 – 1609.
Gao L, Qin GM, Zhou XF. An overview of algorithms for mining frequent patterns in graph data[J]. Acta Electronica Sinica, 2008, 36(8): 1603 – 1609. (in Chinese)
[13] 李先通,安实.基于频繁闭图的图包含查询算法[J].电子学报,2010,38(12):2937 – 2943.
Li XT, An S. A closed frequent subgraph based containment query algorithm[J]. Acta Electronica Sinica, 2010, 38(12): 2937 – 2943. (in Chinese)
[14] Sarkas N, Bansal N, Das G, Koudas N. Measure-driven keyword – query expansion[A]. Proc of the 29th Int'l Conf on Very Large Data Bases[C]. New York: ACM 2005. 121 – 132.
[15] 彭朝晖,崔立真,王珊,张俊,王长亮.一种关系数据库关键词检索相关反馈方法[J].软件学报,2009,20(Supplement):286 – 297.
Peng ZH, Cui LZ, Wang S, Zhang J, Wang CL. Method of relevance feedback in keyword search over relational databases [J]. Journal of Software, 2009, 20(Suppl.): 286 – 297. (in Chinese)
[16] Michael Ley, et al. DBLP Bibliography. [DB/OL] http://www.informatik.uni-trier.de/~ley/db/index.html, 2011-1-15.

作者简介

王新军 男,1968 年生于山东济南.山东大学计算机科学与技术学院教授、博士、博士生导师.主要研究方向为数据库、软件工程、智能数据分析等.

闫 实 男,1986 年生于山东济南.山东大学计算机科学与技术学院硕士研究生.主要研究方向为数据库.

彭朝晖(通信作者) 男,1978 年生于山东临清.山东大学计算机科学与技术学院副教授、博士、硕士生导师.主要研究方向为数据库与信息检索、智能数据分析等.
E-mail: pzh@sdu.edu.cn